CSci 4203 Spring 2019 Lab Assignment 3 :

You are provided a direct mapped cache implemented in SystemVerilog from the companion website for the textbook (http://booksite.elsevier.com/9780124077263/appendices.php) and need to solve the problems described below. SystemVerilog introduces C-like data-types and data-structures which are useful for high-level modelling of computer architecture. Several useful tutorials for SystemVerilog can be found at http://www.asic-world.com/systemverilog which can help you get into it fairly quickly.

Reading :
Read 5.12.1 - 5.12.10 from the online companion website. More specifically, figure 5.12.3 is very useful for understanding the direct mapped cache organization modeled by the given code.

Handout :
A direct-mapped cache is implemented in systemverilog. It uses dm_cache_data.sv, dm_cache_fsm.sv, package_definitions.sv and testbench.sv.

dm_cache_data.sv defines the cache data and tag arrays.
dm_cache_fsm.sv defines the cache controller, which is modelled as an fsm (finite state machine). This file also instantiates the cache data and tag arrays.
package_definitions.sv contains definitions of various data structures and parameters used by the code.
testbench.sv file instantiates the fsm and main memory. Then it tests 10 different kinds of cache accesses. Each testcase causes a request for a cache read or cache write, and prints out the cache and memory accesses to service this request. Use these testcases to understand how the cache works.

Problem 1A (40 points):
Modify the given direct mapped cache to implement a new N-way associative cache, using the LFU (least frequently used) replacement algorithm. The number of sets remains the same (1024 sets), so the N-way associative cache is N times larger than the direct mapped cache. The LFU replacement policies selects the least used cache element for eviction whenever there is a cache miss.

Note :
N is a parameter that can take any value from 1 to 8. This parameter should be defined in ‘package_definitions.sv’. The grader will modify the value of N in this file while testing your code.

Hints **:**

I. Implement LFU(least frequently used) replacement policy by maintaining a counter for each cache line which is incremented when there is a hit and set to 0 when the line is evicted. When there is an eviction request, then the cache line with the smallest counter value will be selected for eviction. If many cache lines have the smallest counter value, then any one of them can be evicted.

II.There are 1024 sets in the direct-mapped cache implementation, with one cache line per set. For an N-way associative cache, the data and tag arrays need to hold 1024xN elements, since there are now 1024 sets and each set has N cache lines.

Problem 1B (10 points) :
Implement an MFU replacement policy (Most Frequently Used). Similar to the LFU policy, maintain a counter for each cache line. The most frequently used entry is evicted.
If multiple entries are used the same number of times, any of them may be evicted.

Problem 2A (50 points) **:**
Change the replacement policy for the set-associative cache to LRU (least recently used).
(30 points)

Hint : Define an array of 1024 queues to keep track of LRU information. Whenever a cache line gets a hit, it will be promoted to the front of the queue corresponding to its set. Whenever there is an eviction, then the cache line at the back of the corresponding queue can be evicted.

Queues :
You can read about queues here : http://www.asic-world.com/systemverilog/data_types14.html

Combinational Loops :
Sometimes, you may find that your code gets stuck in an infinite loop. This might be due to a 'Combinational loop' in your fsm. If two variables in the control logic depend on each other, then they will keep updating their values forever and the simulation will not proceed.

For example :

```
always_comb begin
  a=b;
  b=a;
end
```

Will be stuck forever in a 'combinational loop'.
These can be avoided by ensuring the 'output' and 'input' variables of your combinational block do not have any common variables.

Handin :
Implement your solution by modifying the given *.sv files.
Place your solutions into three folders P1A, P1B and P2.
Do not turn in testbench.sv for any of the problems.

These folders should be placed into another folder, Lab3_StudentName.
Then compress this folder using the command :

```
tar -cvzf Lab3_StudentName.tar.gz Lab3_StudentName
```

Please follow the naming convention closely. For example, do not use p1 or P1_name instead of P1. Please name the compressed folder correctly. It should not be Lab3_StudentName.tar or lab3_StudentName.zip or other variations.

Please submit the tar.gz file following the above steps. The solutions will be auto-graded.
The grader will be provided to you one week after the lab assignment is released.